

1. INTRODUCTION

The increasing number of cars in cities can cause high volume of traffic, and implies that traffic violations become more critical nowadays in Tamil Nadu and also around the world. This causes severe destruction of property and more accidents that may endanger the lives of the people. To solve the alarming problem and prevent such unfathomable consequences, traffic violation detection systems are needed. For which the system enforces proper traffic regulations at all times, and apprehend those who does not comply. A traffic violation detection system must be realized in real-time as the authorities track the roads all the time. Hence, traffic enforcers will not only be at ease in implementing safe roads accurately, but also efficiently; as the traffic detection system detects violations faster than humans. This system can detect traffic light violation in real-time.

1.1 MODULE DESCRIPTION

VIDEO MODULE:

The Traffic Violation and Deduction Module is a web-based tool that helps governments, law enforcement, and other traffic enforcement agencies to effectively manage traffic violations and deductions. This module allows the user to easily enter the details of a violation, assign a penalty, and create a detailed report of the violation and deduction. This software module also helps to keep track of when a payment is due, and it can also generate automated reminders for overdue payments. It also allows for the integration of other systems such as GPS and cameras, to ensure a better understanding of the traffic situation at any given time. This module is a valuable tool for any organization that wishes to effectively manage traffic violations and deductions.

PROCESS MODULE:

This module is designed to process and track traffic violations and deductions. It will automatically login and store information regarding the violation and its associated penalties. It will track the date, time, location, type of violation, and amount of the fine. It will also allow users to view the history of their previous violations and payments, as well as to make payments for any outstanding fines. This module provides a comprehensive system for tracking and processing traffic violations.

REPORTING MODULE:

The Traffic Violation and Deduction Output module is a software program that helps organizations and individuals to accurately identify, track, and report traffic violations and deductions. The module provides detailed information about the violations and deductions, including the type of violation, the severity of the violation, when the violation occurred, and the resulting deductions. The module also provides a summary of all of the violations and deductions for a particular period of time, as well as a comparison between the violations.

2. SYSTEM STUDY

2.1 EXISTING SYSTEM

An existing system for traffic violation and detection typically involves a combination of hardware and software components. Cameras and sensors are strategically placed on the roads and highways to capture images and data on vehicles, drivers, and their behaviours. The data collected is then analysed using software algorithms that can detect and identify a range of traffic violations, including speeding, running red lights, and using mobile phones while driving. Once a violation is detected, the system records the vehicle's license plate number and other relevant information, which is then stored in a database. This database is accessible to law enforcement agencies, who can use the information to issue fines or citations to violators. Some systems may also include automated ticketing capabilities, allowing for immediate issuance of citations without requiring the involvement of law enforcement. These systems can be more efficient and cost-effective than traditional methods, as they reduce the need for human intervention and can process violations more quickly. Overall, traffic violation and detection systems are an important tool for promoting road safety and enforcing traffic laws.

2.2 PROPOSED SYSTEM

The proposed system for traffic violation and detection is a comprehensive system designed to help reduce traffic violations and create a safer environment on the roads. The system provides an efficient way to monitor and detect speed violations. The system involves a combination of features that are designed to help reduce the number of traffic violations and help create a safer driving environment. The system includes a comprehensive database of traffic laws and regulations, which is regularly updated. This database contains information on traffic regulations, such as speed limits and other restrictions. The system also includes a system for monitoring traffic violations, which is able to detect and monitor violations occurring on the roads. The system also includes a penalty system, which is able to apply fines and other penalties for detected violations. The penalties are based on the severity of the violation and may include fines, points on a driver's license, and even suspension of driving privileges. The penalty system is designed to be flexible, allowing the system to adjust the penalty based on the severity of the violation and the circumstances under which the violation occurred. The system also includes a system for tracking and reporting traffic violations.

This system is able to detect and report violations to the appropriate authorities, as well as to the drivers themselves. The system can also be used to track the progress of drivers in regard to traffic violations, allowing the system to adjust the penalties accordingly. Overall, the proposed system for traffic violation and deduction is designed to help reduce traffic violations, create a safer driving environment, and provide drivers with feedback in order to help them improve their driving habits. The system is designed to be flexible and adjust the penalties accordingly, and is able to detect and report violations to the appropriate authorities

2.3 FEASIBILITY STUDY

Potential Impact:

A thorough assessment of the potential impact of the proposed traffic violation and deduction scheme should be conducted. This should include analysing the effect on public safety, the number of traffic violations that may be prevented, the impact on law enforcement resources, the cost of implementing the scheme, and any potential unintended consequences.

Existing Programs:

An examination of existing programs and initiatives aimed at reducing traffic violations should be conducted. This should include assessing the effectiveness of these programs and any potential opportunities for improvement.

Legal Issues:

The legal implications of implementing a traffic violation and deduction scheme should be investigated. This should include examining the existing laws and regulations in the country, as well as any potential conflicts with other laws or regulations.

Cost Analysis:

A comprehensive cost analysis should be conducted to assess the feasibility of implementing the proposed traffic violation and deduction scheme. This should include an examination of both the direct and indirect costs associated with the scheme.

Public Opinion:

A survey or other research should be conducted to assess public opinion regarding the proposed traffic violation and deduction scheme. This should include examining both the level of support and any potential opposition to the scheme.

Implementation:

A plan for the implementation of the traffic violation and deduction scheme should be developed. This should include the timeline for implementation, the roles and responsibilities of all involved parties, the budget and resources needed, and any potential risks or challenges.

Evaluation:

A plan for the evaluation of the scheme should be developed. This should include criteria for assessing the success of the scheme and a timeline for conducting the evaluation. Once all of these aspects have been assessed, a decision can be made regarding the feasibility of the traffic violation and deduction scheme. If the scheme is deemed feasible, then the plan for implementation can be developed and put into action.

3. REQUIREMENT ANALYSIS

Requirement analysis is the process of gathering, documenting, and analysing the needs of a project or system. It is an important step in the software development process and involves understanding the stakeholders' needs, determining the objectives of the project, and determining the scope of the project. It involves analysing the problems, tasks, and constraints of the project and determining the requirements for the system. It also involves understanding the current and future user needs, defining the system objectives, and developing a design for the system. The purpose of requirement analysis is to ensure that the system meets the needs of the stakeholders and that it is designed to meet the objectives of the project.

3.1 BLOCK DIAGRAM

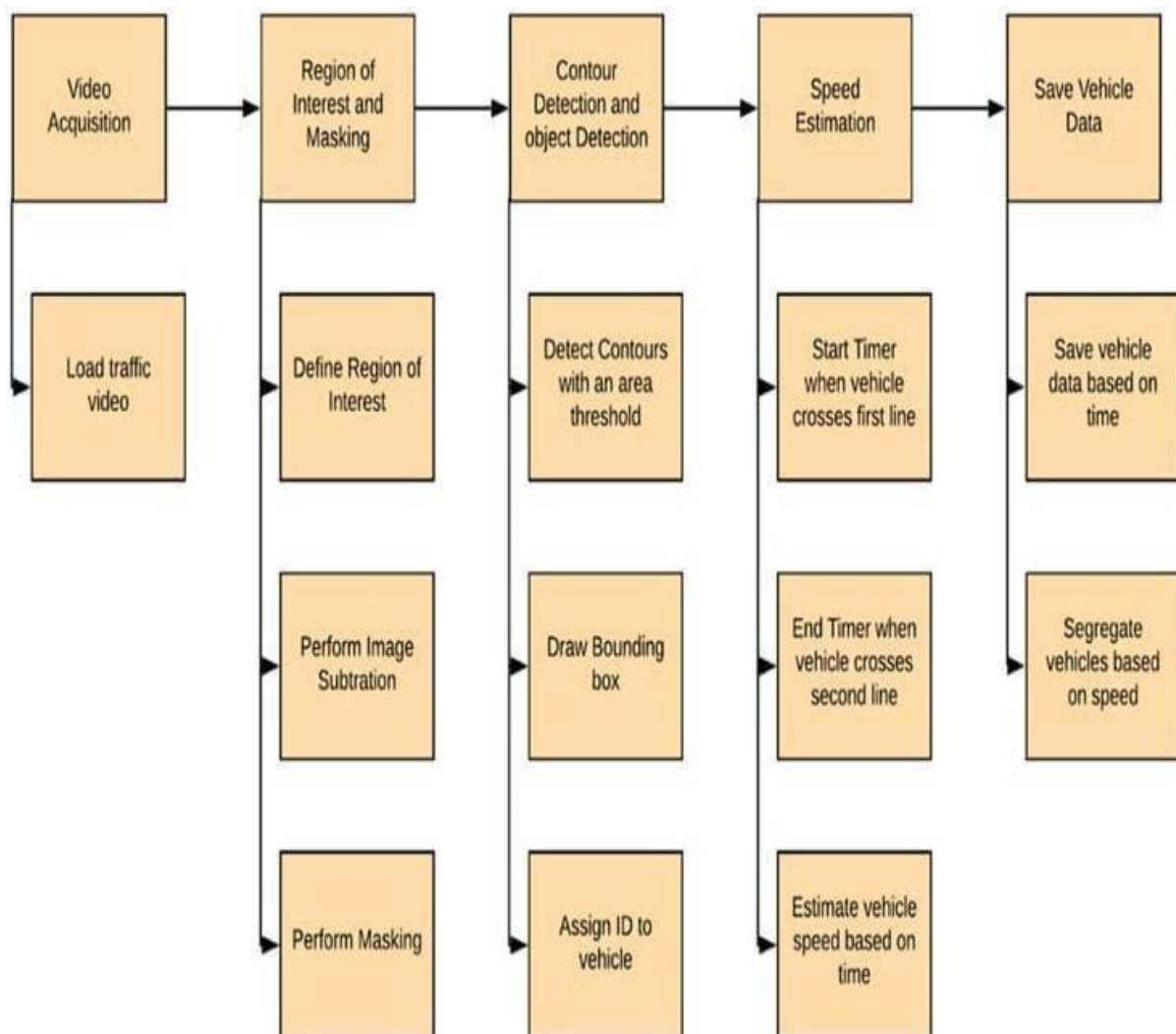


Figure3.1: Block Diagram

4. SYSTEM ANALYSIS

System analysis is the process of analysing a system or process to determine how it should function and what changes should be made to optimize its performance. It is an important part of the overall systems engineering process and involves identifying the system's requirements, how it will work, its components, and the systems that interact with it. System analysis can involve a variety of different methods and techniques, such as flowcharting, interviews, observation, and surveys. The analysis can lead to recommendations for changes or improvements to the system in order to make it more efficient, effective, and reliable. System analysis can also involve problem solving, identifying potential risks, and ensuring compliance with relevant regulations and standards. Ultimately, system analysis is a valuable tool for improving the performance of a system and for ensuring its continued success.

4.1 DATA FLOW DIAGRAM

Data flow diagram (DFD) for "Traffic Violation and Detection" would illustrate the flow of data and information involved in monitoring and detecting traffic violations. The diagram would include entities such as traffic cameras, sensors, and law enforcement agencies, as well as the data and information that is transmitted between these entities. The DFD would also show how data is processed and analysed, such as using computer vision algorithms to detect violations and storing violation records in a database.

Additionally, the diagram would indicate how notifications of violations are sent to law enforcement, as well as how data is shared between different agencies involved in the enforcement of traffic laws. Overall, the DFD would provide a high-level view of the flow of data and information involved in the process of monitoring and detecting traffic violations.

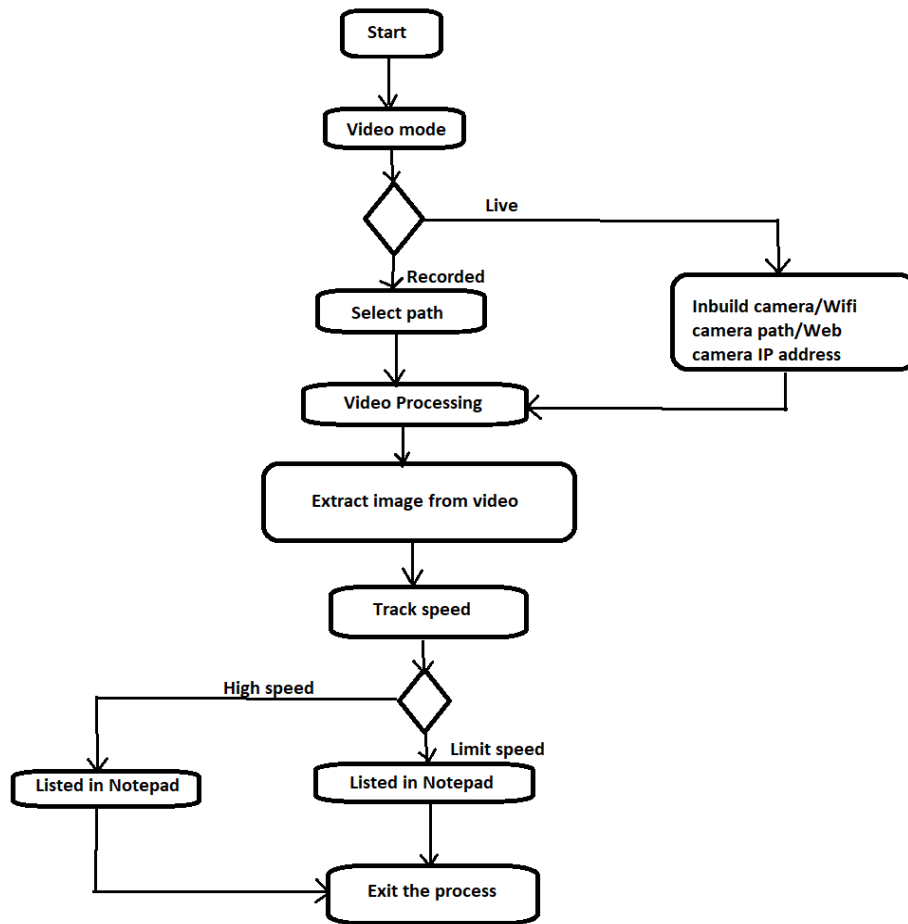


Figure 4.1: Data flow diagram

4.2 ENTITY RELATIONSHIP DIAGRAM

An Entity Relationship Diagram (ERD) is a visual representation of the relationships between different entities in a system. In the context of a traffic violation and deduction system, an ERD diagram would show the various entities involved in the system, such as drivers, vehicles, traffic violations, fines, and point deductions. Overall, an ERD diagram for a traffic violation and deduction system would provide a clear visual representation of the relationships between different entities in the system, helping to ensure that all aspects of the system are properly accounted for and designed to work together efficiently.

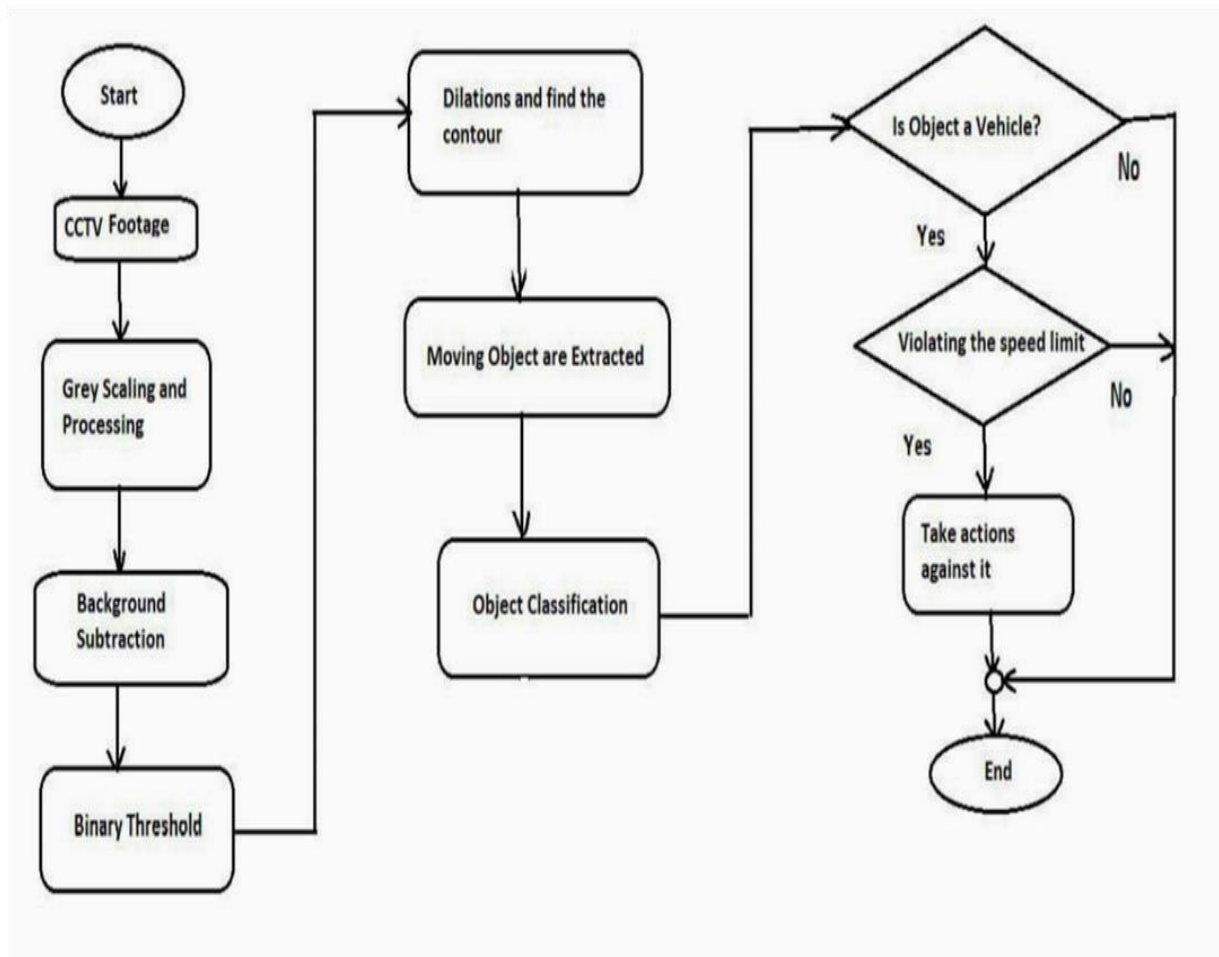


Figure 4.2: Entity Relationship diagram

5. SYSTEM REQUIREMENTS SPECIFICATIONS

5.1 HARDWARE REQUIREMENTS

The hardware requirements for a traffic violation detection project will depend on the scale at which it is implemented. At a minimum, the project will require networked cameras and computer systems to process the video data. For large-scale implementations, a distributed system of cameras and servers may be required. The cameras could be IP cameras, or other cameras with the ability to transmit data over a network. The servers should have enough processing power to run the software required to detect violations. The project may also require other hardware components, such as traffic sensors or radar systems. These components should be connected to the same network as the cameras and servers, so that the data can be collected and processed in real-time. The project may also require storage systems to store the video data and software applications. These storage systems should be large enough to store the data for the period of time required by the project.

Finally, the project may require a display system to show the results of the analysis. This could be an LCD display or a projector connected to the network or an normal system with minimum of 4GB ram and camera system.

Overall, the hardware requirements for a traffic violation detection project will depend on the scale of the project and the type of components used.

5.2 SOFTWARE REQUIREMENTS

Software requirement is the process of identifying and documenting the software functionalities required to implement a system or process. It involves analysing user requirements to determine the desired features of the software and the system environment in which it will be used. In the traffic violation detection project using Python, the software requirement would include the software functionalities to detect the violations and alert the user. This could include features such as facial recognition, license plate recognition, and speed detection. To support these features, the software would need to include an algorithm to process the input data and help detect the violations. Furthermore, the system would need to be integrated with a database to store the data and alert the user. In summary, the software requirement for the traffic violation detection project using Python would include the software functionalities to detect violations and alert the user, a database to store data, and the ability to integrate with other systems.



Figure 5.1: Python logo

The python version we used for this Traffic Violation Detection project is version 3.10.2 which you can see in the below image



Figure 5.2: visual studio code's logo

The idle platform which we are used is Visual Studio Code for developing our Traffic Violation Detection project.

5.2.1 INSTALLED PACKAGES

- CV2
- NUMPY
- TIME
- OS
- TRACKER2
- MATH

1) CV2

Open CV stands for Open-Source Computer Vision (Library). It is the most common and popularly used, well-documented Computer Vision library. Open CV is an open-source library

A screenshot of a Windows Command Prompt window. The title bar says "Command Prompt". The text inside shows the Windows version "10.0.19045.2604", copyright information for Microsoft Corporation, and a command prompt where the user has typed "pip install cv2".

```
Command Prompt
Microsoft Windows [Version 10.0.19045.2604]
(c) Microsoft Corporation. All rights reserved.
C:\Users\LENOVO>pip install cv2
```

that incorporates numerous computer vision algorithms. Open CV increases computational efficiency and assists with real-time applications. One of the major goals of Open CV is to provide an accessible and easy-to-use computer vision infrastructure that helps people build sophisticated computer vision applications quickly.

Figure 5.3: Import the **cv2** package through CMD command.

2) NUMPY

NUMPY is a general-purpose array-processing package. It provides high- performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python Besides its obvious scientific uses, NUMPY can also be used as an efficient multi-dimensional container of generic data.

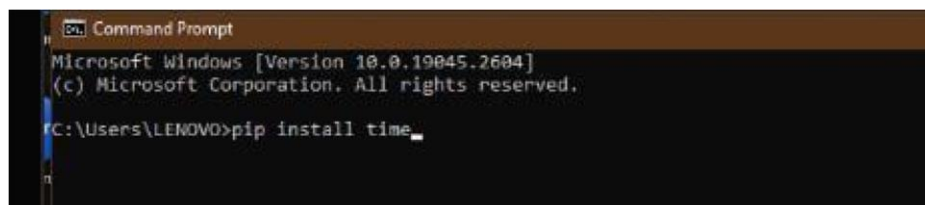


```
Command Prompt
Microsoft Windows [Version 10.0.19045.2604]
(c) Microsoft Corporation. All rights reserved.
C:\Users\LENOVO>pip install numpy_
```

Figure 5.4: Import the **NUMPY** package through CMD command.

3) TIME

Date Time objects represent instants in time and provide interfaces for controlling its representation without affecting the absolute value of the object. Date Time objects may be created from a wide variety of string or numeric data, or may be computed from other Date Time objects. Date Times support the ability to convert their representations to many major time zones, as well as the ability to create a Date Time object in the context of a given time zone.



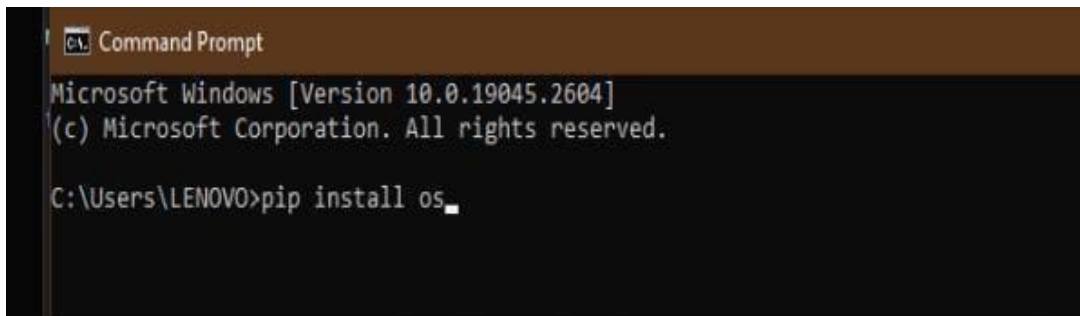
```
Command Prompt
Microsoft Windows [Version 10.0.19045.2604]
(c) Microsoft Corporation. All rights reserved.
C:\Users\LENOVO>pip install time_
```

Figure 5.5: Import the **Date time** package through CMD command

4) OS

Python OS module provides the facility to establish the interaction between the user and the operating system. It offers many useful OS functions that are used to perform OS-based tasks and get related information about operating system. The OS comes under Python's standard utility modules. This module offers a portable way of using operating system dependent functionality.

The Python OS module lets us work with the files and directories of the system.

A screenshot of a Windows Command Prompt window. The title bar is dark blue with the text 'Command Prompt'. The window content shows the following text: 'Microsoft Windows [Version 10.0.19045.2604]', '(c) Microsoft Corporation. All rights reserved.', and 'C:\Users\LENOVO>pip install os'. The cursor is at the end of the command line.

```
Microsoft Windows [Version 10.0.19045.2604]
(c) Microsoft Corporation. All rights reserved.

C:\Users\LENOVO>pip install os
```

Figure 5.6: import the OS package through CMD command

6. SYSTEM DESIGN

6.1 INTERFACE DESIGN:

Traffic violation detection is a critical issue in the modern world. It is a major factor in road safety and is essential for maintaining the flow of traffic. It is equally important for the safety of pedestrians, cyclists, and other road users. This project seeks to develop an interface design for a traffic violation detection system. The system will be used to detect, record, and report traffic violations in real-time.

6.2 SYSTEM OVERVIEW:

The system will be composed of a set of traffic cameras connected to a central server. The cameras will be used to detect and record traffic violations. They will be equipped with the necessary hardware and software to detect and record the various types of violations. The central server will be responsible for processing the data and providing information to law enforcement agencies. It will also provide an interface to the public, allowing them to view traffic violations in their area.

6.3 USER INTERFACE DESIGN:

The user interface of the traffic violation detection system will be designed using a modern web-based framework. The interface will be divided into two main sections: the main page and the violation report page.

6.4 MAIN PAGE:

The main page will be the main entry point of the system. It will display a map of the area covered by the system with icons indicating the location of the cameras. It will also provide information about the type of violations detected by each camera.

6.5 VIOLATION REPORT:

The violation report page will provide detailed information about violations that have been detected. It will include a list of violations, their location, and the time of the violation. It will also provide options to print or download the report.

6.6 IMPLEMENTATION OF THE SYSTEM:

The system will be implemented using a cloud-based platform. The platform will provide the necessary infrastructure to support the traffic violation detection system. It will also provide a user interface for the system. The platform will be responsible for providing the necessary security measures to ensure the system is secure.

6.7 CONCLUSION:

This project has outlined a user interface design for a traffic violation detection system. The system will be composed of a set of cameras connected to a central server. The user interface will be designed using a modern web-based framework and will provide a map of the area covered by the system and detailed information about violations that have been detected. The system will be implemented using a cloud-based platform to provide the necessary infrastructure and security measures.

7. SCREENSHOTS

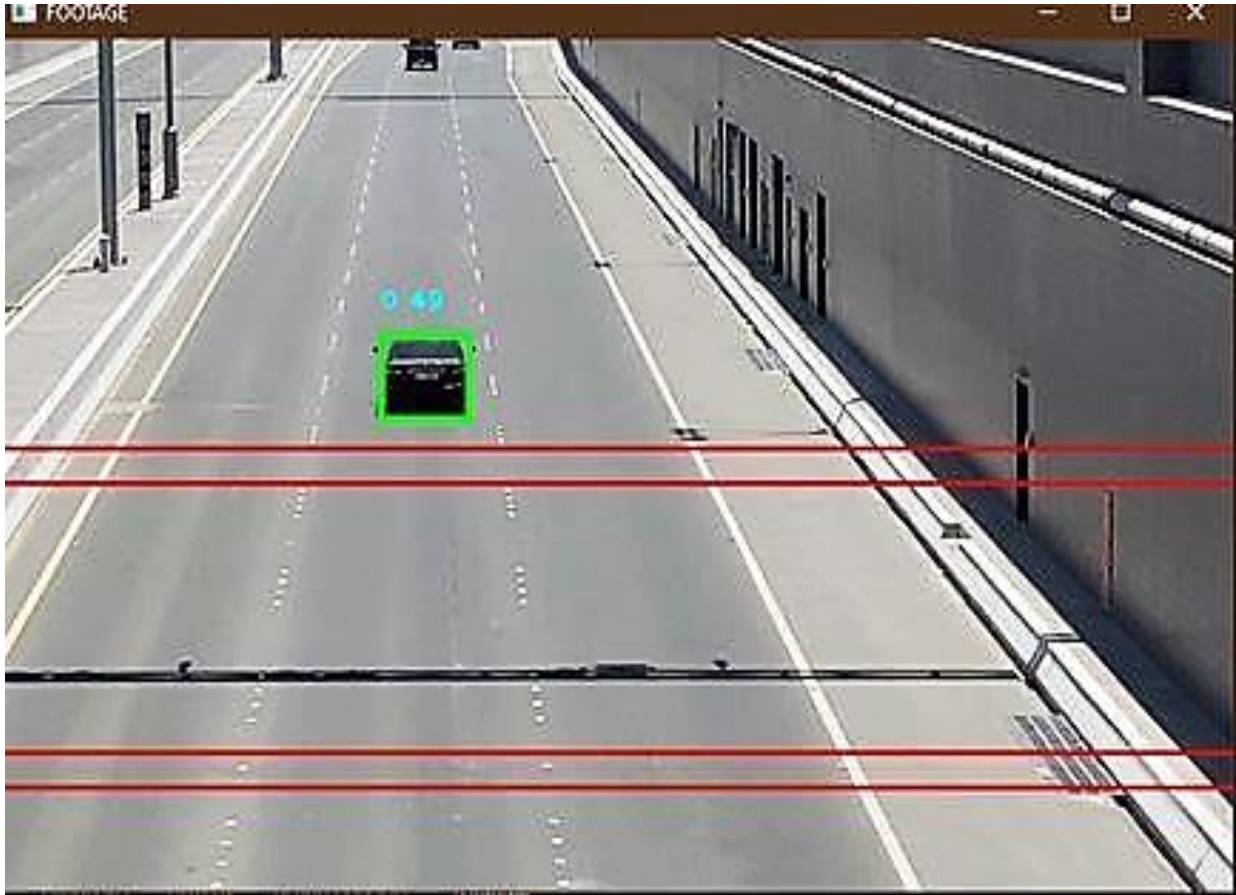


Figure 7.1: Drawing lanes for vehicle detection

As you can see the above image represents the output of the footage where the lines are used to detect the vehicle which are all passing through the particular lane. Once the vehicle touches the starting line the process will take place, where it will start initializing the speed and id of the vehicle. It uses background subtraction process, where we are able to track down the object in accurate manner. Once those processes are over, the program starts to initialize and calibrate the speed of the vehicle.

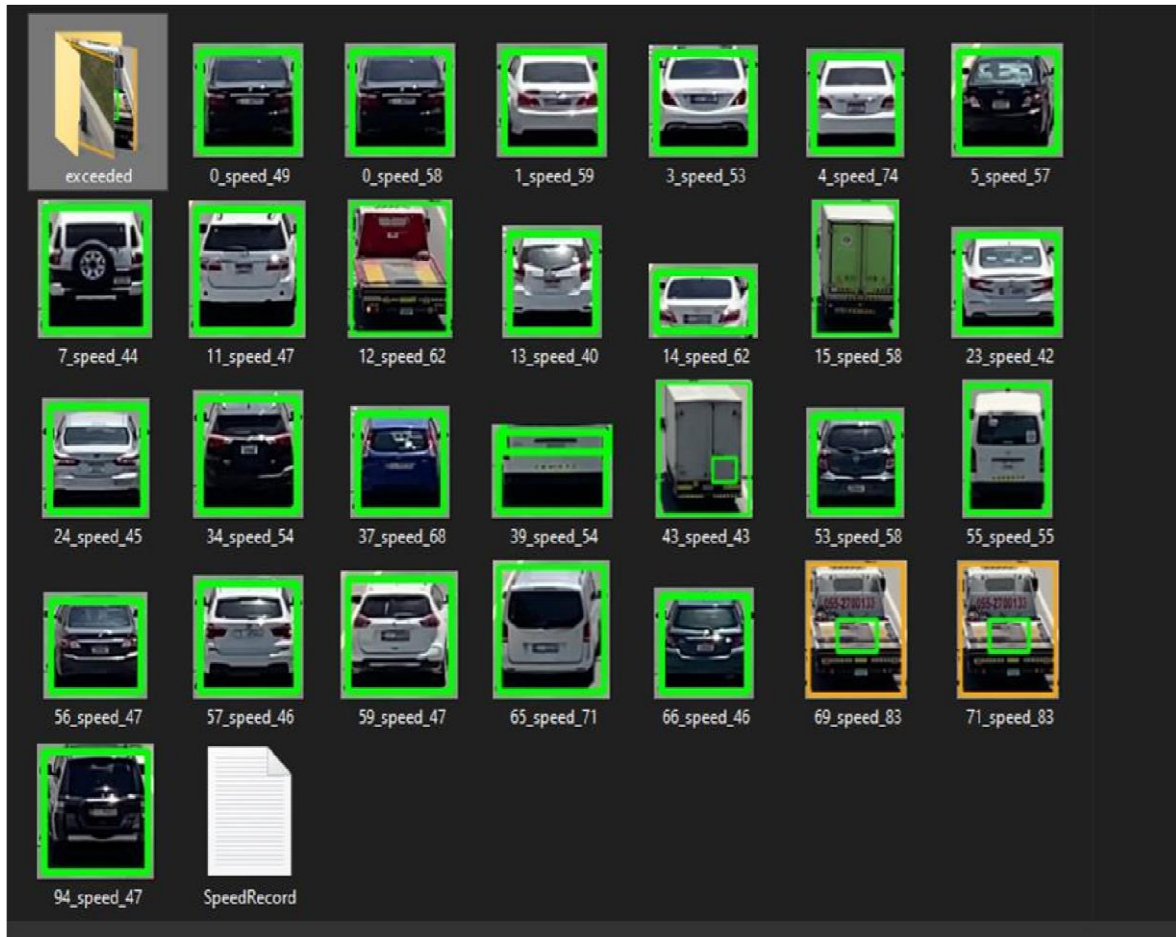


Figure 7.2: Captured images are saved in separate file

The above image represents the data storage of the output images where we took each and every vehicle which has passed through the particular lane's line. As you can see there will be two different folders to store the vehicle based on the speed. Once the vehicle passed the lanes, the program will be calibrate the speed of the vehicle and classify that the car or the truck has crossed the lane in the correct speed or not. In this program the fixed speed is 80 .Whenever the vehicle passes the lane, the vehicle classified was classified and recorded according to it speed, as you can see in the above image. When the vehicle is average are less than 80 KMH it will be noted and marked in green colour box .whereas the orange box indicates that the vehicle speed exceeded the limit.



Figure 7.3: Green box indicates correct speed

The above image represents the vehicle pictures which are all not exceeded or not violated the speed limit. So that they are bordered in green colour.

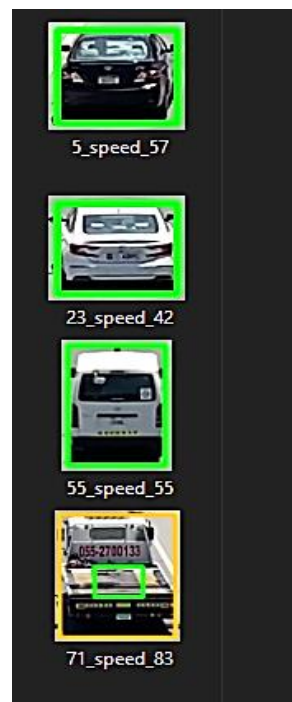


Figure 7.4: speed violated vehicle highlighted in orange squares

In the **figure 7.4** we can see that, those who violated the speed limit will be captured in a different colour so that we can able to easily rectify the vehicle's image easily.

```
ID      SPEED OF THE VECHICLE
-----
0       49
-----
SUMMARY
-----
Total Vehicles :      1
Exceeded speed limit :    0
```

Figure 7.5: Saved vehicle records in notepad text editor

In the above image we can see a text editor, which shows the records the vehicle which has been passed through the end of the footage with their id and speed .In this process we can get a clear view of the vehicles list where in the video footage.

8. SAMPLE CODING

8.1 Video Acquisition

```
cap = cv2.VideoCapture("project file")
```

Region of Interest and Masking

```
#KERNALS
```

```
kernalOp = np.ones((3,3),np.uint8) kernalOp2 = np.ones((5,5),np.uint8) kernalCl =  
np.ones((11,11),np.uint8)
```

```
fgbg=cv2.createBackgroundSubtractorMOG2(detectShadows=True)
```

```
#MASKING
```

```
fgmask = fgbg.apply(roi) ret, imBin = cv2.threshold(fgmask, 200, 255,  
cv2.THRESH_BINARY) mask1 = cv2.morphologyEx(imBin, cv2.MORPH_OPEN,  
kernalOp) mask2 = cv2.morphologyEx(mask1, cv2.MORPH_CLOSE, kernalCl)
```

8.2 Contour detection:

```
contours,=cv2.findContours(mask2,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
```

```
detections = [] for cnt in contours:
```

```
#print(cnt)          area      =
```

```
cv2.contourArea(cnt)  if area >
```

```
1000:
```

```
x,y,w,h=cv2.boundingRect(cnt)cv2.rectangle(roi,(x,y),(x+w,y+h),(0,255,0),3)
```

```
detections.append([x,y,w,h])
```

8.3 Object Tracking

```
for rect in objects_rect: x, y,
w, h = rect cx =
(x + x + w) // 2 cy =
(y + y + h) // 2

# Find out if that object was detected already

same_object_detected= False for id, pt in self.center_points.items():

dist = math.hypot(cx - pt[0], cy - pt[1])

if dist < 150:

self.center_points[id] = (cx, cy) #print(self.center_points)
objects_bbs_ids.append([x, y, w, h, id]) same_object_detected
= True
```

8.4 Speed Estimation and Timer Start and stop

```
if (y >= 325 and y <= 345): self.s1[id] =
time.time() if (y >= 150 and y <= 170):
self.s2[id] = time.time() self.s[id] =

def getspeed(self,id):

if

(self.s[id]!=0):

s = 439.8/self.s[id]

else: s = 0

return s
```

8.5 Drawing Rectangles and displaying on the screen for box_id:

```
x,y,w,h,id = box_id
```

```
cv2.putText(roi,str(id)+"any parameter"+str(tracker.getsp(id)),(x,y-15), cv2.FONT_HERSHEY_PLAIN,1,(255,255,0),2)
```

```
#print(tracker.getsp(id)) cv2.rectangle(roi, (x, y), (x + w, y + h),  
(0, 255, 0), 3)
```

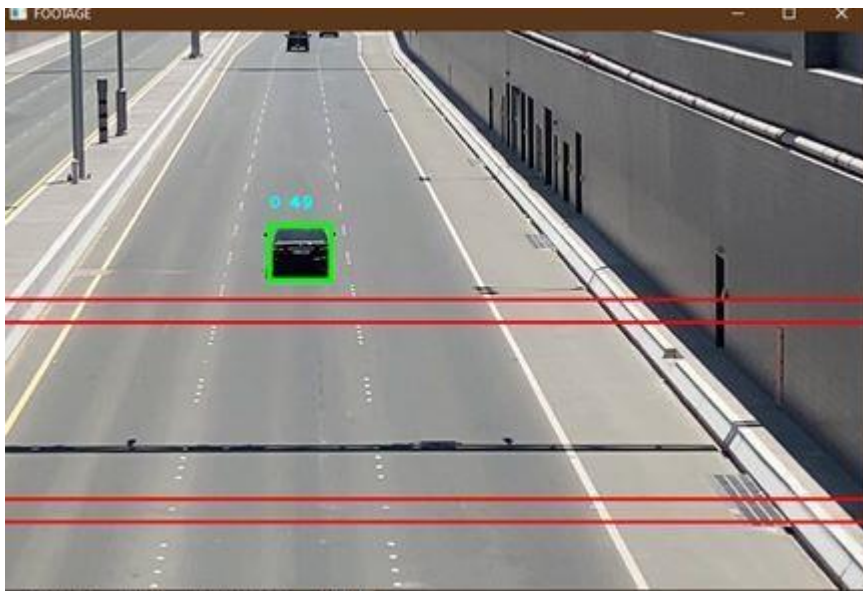


Figure 8.1: Code to detect the speed of the vehicle

8.6 Drawing Reference Lines

```
cv2.line(roi, (0, 325), (960, 325), (0, 0, 255), 2)
```

```
cv2.line(roi, (0, 345), (960, 345), (0, 0, 255), 2)
```

```
cv2.line(roi, (0, 150), (960, 150), (0, 0, 255), 2)
```

```
cv2.line(roi, (0, 170), (960, 170), (0, 0, 255), 2)
```

```
#choose the parameter based on the video frame which we are  
importing
```

8.7 Save Vehicle's Images and speed

```
def
capture(0): if(self.capf[id]==0): self.capf[id] = 1 self.f[id]=0
crop_img = img[y- 5:y + h+5, x-5:x + w+5] n =
str(id)+"_speed_"+str(sp) file = 'D://TrafficRecord/' + n + '.jpg'
cv2.imwrite(file, crop_img) self.count += 1 file1 =
open("D://TrafficRecord//SpeedRecord.txt", "a") if(sp>limit):

file2 = 'D://TrafficRecord//exceeded/' + n + '.jpg' cv2.imwrite(file2, crop_img)
file1.write(str(id)+" \t "+str(sp)+"<---exceeded\n") self.exceeded+=1 else:

file1.write(str(id) + " \t " + str(sp) + "\n") file1.close()
```

Above code is used to save the picture of the vehicle and store in the projects

. 8.8 Code for speed detecting

```
import cv2

from tracker2 import *

import numpy as np

end = 0

tracker = EuclideanDistTracker()

cap = cv2.VideoCapture('traffic4.mp4')

f = 25

w = int(1000/(f-1))

#Object Detection
```

```

object_detector =
cv2.createBackgroundSubtractorMOG2(history=None,varThreshold=None)

#100,5

#KERNALS

kernalOp = np.ones((3,3),np.uint8)

kernalOp2 = np.ones((5,5),np.uint8)

kernalCl = np.ones((11,11),np.uint8)

fgbg=cv2.createBackgroundSubtractorMOG2(detectShadows=True)

kernal_e = np.ones((5,5),np.uint8)

while True:

ret,frame = cap.read()

if not ret:

break

frame = cv2.resize(frame, None, fx=0.5, fy=0.5)

height,width,_ = frame.shape

#print(height,width)

#Extract ROI

roi = frame[50:540,200:960]

#MASKING METHOD 1

mask = object_detector.apply(roi)

```



```

_, mask = cv2.threshold(mask, 250, 255, cv2.THRESH_BINARY)

#DIFFERENT MASKING METHOD 2 -> This is used

fgmask = fgbg.apply(roi)

ret, imBin = cv2.threshold(fgmask, 200, 255, cv2.THRESH_BINARY)

mask1 = cv2.morphologyEx(imBin, cv2.MORPH_OPEN, kernalOp)

mask2 = cv2.morphologyEx(mask1, cv2.MORPH_CLOSE, kernalCl)

e_img = cv2.erode(mask2, kernal_e)

contours,_ = cv2.findContours(e_img,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)

detections = [ ]

for cnt in contours:

    area = cv2.contourArea(cnt)

#THRESHOLD

if area > 1000:

    x,y,w,h = cv2.boundingRect(cnt)

    cv2.rectangle(roi,(x,y),(x+w,y+h),(0,255,0),3)

    detections.append([x,y,w,h])

#Object Tracking

boxes_ids = tracker.update(detections)

for box_id in boxes_ids:

    x,y,w,h,id = box_id

```

```

if(tracker.getsp(id)<tracker.limit()):

cv2.putText(roi,str(id)+""+str(tracker.getsp(id)),(x,y-15),
cv2.FONT_HERSHEY_PLAIN,1,(255,255,0),2)

cv2.rectangle(roi, (x, y), (x + w, y + h), (0, 255, 0), 3)

else:

cv2.putText(roi,str(id)+ " "+str(tracker.getsp(id)),(x, y-
15),cv2.FONT_HERSHEY_PLAIN, 1,(0, 0, 255),2)

cv2.rectangle(roi, (x, y), (x + w, y + h), (0, 165, 255), 3)

s = tracker.getsp(id)

if (tracker.f[id] == 1 and s != 0):

tracker.capture(roi, x, y, h, w, s, id)

# DRAW LINES

cv2.line(roi, (0, 410), (960, 410), (0, 0, 255), 2)

cv2.line(roi, (0, 430), (960, 430), (0, 0, 255), 2)

cv2.line(roi, (0, 235), (960, 235), (0, 0, 255), 2)

cv2.line(roi, (0, 255), (960, 255), (0, 0, 255), 2)

#DISPLAY

#cv2.imshow("Mask",mask2)

cv2.imshow("FOOTAGE", roi)

if cv2.waitKey(20) & 0xFF == ord('q'):

```

```
break
```

```
if(end!=1):
```

```
tracker.end()
```

```
cap.release()
```

```
cv2.destroyAllWindows()
```

8.9 Code for speed detecting

```
import cv2
```

```
import math
```

```
import time
```

```
import numpy as np
```

```
import os
```

```
limit = 80 # km/hr
```

```
traffic_record_folder_name = "TrafficRecord"
```

```
if not os.path.exists(traffic_record_folder_name):
```

```
    os.makedirs(traffic_record_folder_name)
```

```
    os.makedirs(traffic_record_folder_name+"//exceeded")
```

```
speed_record_file_location = traffic_record_folder_name + "//SpeedRecord.txt"
```

```
file = open(speed_record_file_location, "w")
```

```
file.write("ID \t SPEED OF THE VECHICLE\n-----\t-----\n")
```

```
file.close()
```

```

class EuclideanDistTracker:

    def __init__(self):

        # Store the center positions of the objects

        self.center_points = {}

        self.id_count = 0

        # self.start = 0

        # self.stop = 0

        self.et = 0

        self.s1 = np.zeros((1, 1000))

        self.s2 = np.zeros((1, 1000))

        self.s = np.zeros((1, 1000))

        self.f = np.zeros(1000)

        self.capf = np.zeros(1000)

        self.count = 0

        self.exceeded = 0

    def update(self, objects_rect):

        objects_bbs_ids = []

```

```

# Get center point of new object

for rect in objects_rect:

    x, y, w, h = rect

    cx = (x + x + w) // 2

    cy = (y + y + h) // 2


# CHECK IF OBJECT IS DETECTED ALREADY

same_object_detected = False

for id, pt in self.center_points.items():

    dist = math.hypot(cx - pt[0], cy - pt[1])

    if dist < 70:

        self.center_points[id] = (cx, cy)

        objects_bbs_ids.append([x, y, w, h, id])

        same_object_detected = True


# START TIMER

if (y >= 410 and y <= 430):

    self.s1[0, id] = time.time()


# STOP TIMER and FIND DIFFERENCE

if (y >= 235 and y <= 255):

    self.s2[0, id] = time.time()

```

```

        self.s[0, id] = self.s2[0, id] - self.s1[0, id]

    # CAPTURE FLAG

    if (y < 235):

        self.f[id] = 1

    # NEW OBJECT DETECTION

    if same_object_detected is False:

        self.center_points[self.id_count] = (cx, cy)

        objects_bbs_ids.append([x, y, w, h, self.id_count])

        self.id_count += 1

        self.s[0, self.id_count] = 0

        self.s1[0, self.id_count] = 0

        self.s2[0, self.id_count] = 0

    # ASSIGN NEW ID to OBJECT

    new_center_points = {}

    for obj_bb_id in objects_bbs_ids:

        _, _, _, _, object_id = obj_bb_id

        center = self.center_points[object_id]

        new_center_points[object_id] = center

    self.center_points = new_center_points.copy()

```

```

        return objects_bbs_ids

# SPEEED FUNCTION

def getsp(self, id):

    if (self.s[0, id] != 0):

        s = 214.15 / self.s[0, id]

    else:

        s = 0

    return int(s)

# SAVE VEHICLE DATA

def capture(self, img, x, y, h, w, sp, id):

    if (self.capf[id] == 0):

        self.capf[id] = 1

        self.f[id] = 0

        crop_img = img[y - 5:y + h + 5, x - 5:x + w + 5]

        n = str(id) + "_speed_" + str(sp)

        file = traffic_record_folder_name + '/' + n + '.jpg'

        cv2.imwrite(file, crop_img)

        self.count += 1

        filet = open(speed_record_file_location, "a")

        if (sp > limit):

```

```

file2 = traffic_record_folder_name + '//exceeded//' + n + '.jpg'

cv2.imwrite(file2, crop_img)

file.write(str(id) + " \t " + str(sp) + "<---exceeded\n")

self.exceeded += 1

else:

    file.write(str(id) + " \t " + str(sp) + "\n")

file.close()

# SPEED_LIMIT

def limit(self):

    return limit

# TEXT FILE SUMMARY

def end(self):

    file = open(speed_record_file_location, "a")

    file.write("\n-----\n")

    file.write("-----\n")

    file.write("SUMMARY\n")

    file.write("-----\n")

    file.write("Total Vehicles :\t" + str(self.count) + "\n")

    file.write("Exceeded speed limit :\t" + str(self.exceeded))

    file.close()

```


9. TESTING

9.1 TEST PROCESS

The traffic violation and detection process consists of four main steps: detection, notification, enforcement, and deterrence.

9.1.2 Detection:

The first step in the process is detection. This involves the use of cameras and other technologies to detect traffic violations. For example, cameras can be used to detect speeding, running red lights, and other violations. In addition, some systems use sensors to detect when vehicles have stopped too long at a stop sign or traffic light.

9.1.3 Notification:

Once a violation has been detected, the system will send out a notification to the appropriate law enforcement agency. This notification will include a description of the violation, the location of the violation, and the time of the violation. This information is then used to issue a citation to the driver.

9.1.4 Enforcement:

Once a citation has been issued, the enforcement process begins. This process involves the use of fines, points on the driver's license, and possible imprisonment, depending on the severity of the violation.

9.1.5 Deterrence:

The goal of the deterrence process is to prevent future violations. This is done through a variety of methods, including public service announcements, educational campaigns, and increased fines and punishments for violations. Additionally, some systems are designed to detect violations before they occur and alert drivers to the potential danger.

Overall, the traffic violation and detection process is an important part of keeping roads and highways safe. By using cameras and other technologies to detect violations, law enforcement is able to more effectively enforce the laws and deter drivers from committing violations. This process is essential in ensuring that all drivers follow the rules of the road and remain safe.

9.2 TEST CASES

The traffic violation and detection project is used to detect traffic violations and to ensure that drivers obey the speed limits. The project includes the use of computer vision techniques to detect traffic violations and enforce speed limits. The system can also be used to collect data on traffic patterns, identify potential sources of congestion and provide feedback to road users. The aim of this test-case design is to ensure that the system is able to detect traffic violations and enforce speed limits accurately and efficiently.

9.2.1 Test Objectives:

The main objective of the test-case design is to verify the accuracy of the system in detecting traffic violations and enforcing speed limits. The test should also be able to identify potential sources of congestion and provide feedback to road users.

9.2.2 Test Plan:

The test plan for the traffic violation and detection project will involve the following steps:

1. **System Quality Assurance:** The first step of the test plan is to ensure the quality of the system. The test should verify that the system is designed and implemented to the highest standards and is able to detect traffic violations and enforce speed limits accurately and efficiently.

2. **System Validation:** The second step of the test plan is to validate the system. The test should verify that the system is able to detect traffic violations and enforce speed limits accurately and efficiently.

3. System Performance Evaluation: The third step of the test plan is to evaluate the performance of the system. The test should be able to identify potential sources of congestion and provide feedback to road users.

4. System Usability Evaluation: The fourth step of the test plan is to evaluate the usability of the system. The test should verify that the system is user-friendly and easy to use.

9.2.3 Test Cases:

Test Case 1: This test case is designed to check the accuracy of the system in detecting traffic violations and enforcing speed limits. The test should verify that the system is able to detect traffic violations and enforce speed limits accurately and efficiently.

Test Case 2: This test case is designed to check the accuracy of the system in detecting potential sources of congestion. The test should verify that the system is able to identify potential sources of congestion accurately and efficiently.

Test Case 3: This test case is designed to check the accuracy of the system in providing feedback to road users. The test should verify that the system is able to provide feedback to road users accurately and efficiently.

Test Case 4: This test case is designed to check the usability of the system. The test should verify that the system is user-friendly and easy to use.

9.2.4 Conclusion:

The test-case design for the traffic violation and detection project is designed to ensure the accuracy, performance and usability of the system. The test should verify that the system is able to detect traffic violations and enforce speed limits accurately and efficiently. The test should also be able to identify potential sources of congestion and provide feedback to road users. The test should also be able to verify that the system is user-friendly and easy to use.

The test-case design is designed to ensure the quality and reliability of the system.

9.2.5 Testing Types:

The development of a traffic violation and detection system is a complex and multi-faceted process. It requires careful consideration of the components that are necessary to

develop a successful system. The primary components of a traffic violation and detection system include the hardware, software, data, and processes. This paper will discuss the various types of testing that must be performed to ensure the successful implementation of such a system. This paper will also discuss the importance of each type of testing and the potential benefits of each.

Hardware testing is the process of validating the hardware components of the system. This includes testing the accuracy and reliability of the sensors, cameras, and other hardware components. Additionally, it involves testing the functionality of the hardware components to ensure that they are working as expected. This type of testing is necessary to ensure that the system is able to accurately detect and record violations.

Software testing is the process of validating the software components of the system. This includes testing the accuracy and reliability of the algorithms and other software components.

It also involves testing the usability of the software to ensure that it is easy to use and understand. Additionally, software testing is used to detect any potential security flaws in the software. This type of testing helps to ensure that the system is secure and can detect violations accurately.

Data testing is the process of validating the data that is used by the system. This includes testing the accuracy and reliability of the data as well as testing the integrity of the data. Additionally, it involves testing the data for any anomalies or errors. This type of testing is necessary to ensure that the system is able to accurately detect and record violations.

Process testing is the process of validating the processes that are used by the system. This includes testing the accuracy and reliability of the algorithms, processes, and other components. Additionally, it involves testing the usability of the processes to ensure that they are easy to use and understand. This type of testing helps to ensure that the system is able to accurately and efficiently detect and record violations.

Usability testing is the process of validating the user interface of the system. This includes testing the accuracy and reliability of the user interface components. Additionally, it involves testing the usability of the user interface to ensure that it is easy to use and understand.

This type of testing helps to ensure that the system is able to accurately detect and record violations.

Performance testing is the process of validating the system's performance. This includes testing the accuracy and reliability of the system's performance. Additionally, it involves testing the system's performance under varying conditions to ensure that it is able to accurately detect and record violations. This type of testing is necessary to ensure that the system is able to accurately detect and record violations in a timely manner.

Overall, there are several types of testing that must be performed in order to ensure the successful implementation of a traffic violation and detection system. Each type of testing is important in ensuring that the system is able to accurately detect and record violations. Additionally, it helps to ensure that the system is secure and able to provide reliable results. Therefore, it is important to ensure that all types of testing are performed in order to ensure the successful implementation of a traffic violation and detection system.

10. CONCLUSION

The traffic violation detection project is a useful and innovative way to reduce traffic violations and accidents caused by careless drivers. It helps traffic enforcement agencies to identify violators and take necessary legal actions against them, while providing valuable data to understand traffic flow and take corrective actions. The project allows for remote monitoring and provides real-time information to drivers and road users about violations, creating awareness about road safety. The technology used is cutting-edge, cost-effective, and highly scalable, making it an invaluable tool to reduce traffic violations and create a safer road environment.

11. REFERENCES

- [1] G. Ou, Y. Gao and Y. Liu, "Real Time Vehicular Traffic Violation Detection in Traffic Monitoring System," in 2012 IEEE/WIC/ACM, Beijing, China , 2012.
- [2] X. Wang, L.-M. Meng, B. Zhang, J. Lu and K.-L. Du, "A Video-based Traffic Violation Detection System," in MEC, Shenyang, China, 2013.
- [3] Joseph Redmon and Ali Farhadi, "YOLOv3: An Incremental Improvement".
- [4] ScienceDirect – 2020 - Determining Vehicle Speeds Using Convolutional Neural Networks by Alexander Grents, Vitalii Varkentin*, Nikolay Goryaev
- [5] ScienceDirect – 2020 - Detection of Vehicle Position and Speed using Camera Calibration and Image Projection Methods by Alexander A S Gunawana,* , Deasy Aprilia Tanjunga, Fergyanto E. Gunawan
- [6] ScienceDirect – 2020 - An Efficient Approach for Detection and Speed Estimation of Moving Vehicles By Tarun Kumar and Dharmender Singh Kushwaha
- [7] ScienceDirect – 2020 - Vehicle speed measurement model for video-based systems by Saleh Javadi , Mattias Dahl, Mats I. Pettersson .